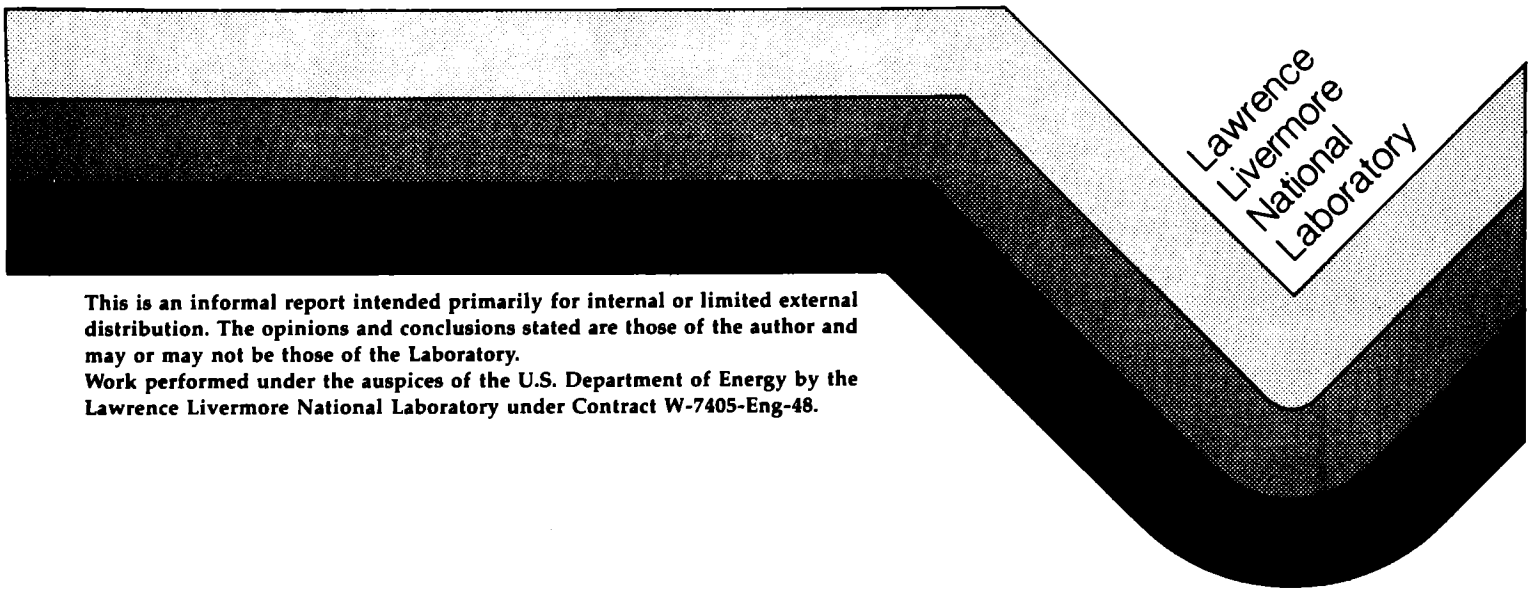
UCID-21126

INTRODUCTION AND GUIDE TO LLNL'S RELATIVISTIC 3-D NUCLEAR HYDRODYNAMICS CODE

ZINGMAN, JONATHAN A.
MCABEE, THOMAS L.
ALONSO, CAROL T.
WILSON, JAMES R.

NOVEMBER 1987



Work performed under the auspices of the U.S. Department of Energy by the Lawrence Livermore National Laboratory under Contract W-7405-Eng-48.

DISCLAIMER

Printed in the United States of America
Available from
National Technical Information Service
U.S. Department of Commerce
5285 Port Royal Road
Springfield, VA 22161

Price Code	Page Range
A01	Microfiche
Papercopy Prices	
A02	001-050
A03	051-100
A04	101-200
A05	201-300
A06	301-400
A07	401-500
A08	501-600
A09	601

The Bevalac has provided opportunities for nuclear experimentalists by opening higher energy and mass ranges than had previously been accessible. Greater energy meant that relativistic effects would now be important and greater mass that new collective effects could show up. Both of these expectations have been borne out and, consequently, theories to describe the physics in this regime must address these experiments.

The way to ensure that relativistic effects are correctly dealt with is to use a covariant theory, preferably with a well understood low-velocity limit. The necessity for this treatment is becoming evident in low energy nuclear structure physics, as recent successes of Dirac phenomenology have shown.[1] Theories which have very few adjustable parameters, but whose form is dictated by Lorentz invariance, have proved to describe data well.[2] The failure of nonrelativistic, collective theories to predict single-particle-inclusive cross sections[3] also indicates that the correct treatment of special relativity is important.

It is less clear how collective effects such as bulk flow of matter arise. Single-particle models are not likely to be useful, since the number of degrees of freedom in relativistic heavy-ion collisions is quite large. Quantum mechanical models are also problematical due to the strong interactions between particles and the many-body nature of the colliding systems. Hence, classical statistical models of some sort are the best choice for describing the physics phenomenologically in this regime.

Work on collective models has proceeded in two directions. The first is to approach the problem from kinetic theory. In such a description, the colliding nuclei are viewed as collections of particles that interact either pairwise or propagate in a mean field. Generally, an equation such as the Boltzmann equation[4] or the VUU equation[5] is integrated by following an ensemble of test particles. From this, a momentum distribution function is derived which eventually yields the experimental observables. Since the interactions treated are essentially two-body in

nature, collective effects do not arise in an obvious fashion from these equations. Including corrections such as the bulk equation of state[6] helps to explain some of the observed collective behavior.

The other approach to the collision dynamics is through hydrodynamic models. Nonrelativistic hydrodynamics can be derived from the Schrödinger equation by separating the wave function into its real and imaginary parts and taking the limit $\hbar \to 0$. Many ideal and nonideal, nonrelativistic hydrodynamic models have been proposed, but none of them adequately explains the experimental data.[3] Relativistic hydrodynamic models in up to three spatial dimensions also have been developed[7] and have had somewhat greater success.

Due to the above considerations, we have constructed a relativistic hydrodynamic model to investigate Bevalac and higher energy, heavy-ion collisions. The basis of the model is a finite-difference solution to covariant hydrodynamics, which will be described in the rest of this paper. Consideration of hydrodynamic flow alone is insufficient for useful nuclear predictions. Pions are generated with the available energy in the collision, and the nuclei break up after impact. We are treating these topics as well and will describe our methods in papers following this one.[8]

This paper is organized as follows. A brief review of the equations and numerical methods we have employed in the solution to the hydrodynamic equations forms Section II. Section III is a detailed description of several of the most important subroutines. In Section IV, we present numerical tests on the code. Finally, Section V has our conclusions and prospects for future work. We also provide two appendices. The first details running the program, while the second describes the output which the program currently provides.

SECTION II

Due to the extensive experience at Lawrence Livermore National Laboratory with finite difference methods for solution of the hydrodynamic equations, we chose to use these methods for our model. In practice, this means that we finite difference fields that are centered on a mesh rather than propagate marker particles carrying momentum and energy through the grid, as is done in particle in-cell methods.[7] Our method has a number of advantages, among them being that, at least for simple cases, the scheme can be shown explicitly to have second-order accuracy in time and space. Due to the possible presence of shocks from the impact, this high accuracy is desirable. Since the model is currently implemented on CRAY computers with vectorizing compilers, the cost for second-order accuracy is not prohibitive.

A general outline of the equations and methods used can be found in Ref. 9. There, general relativistic two-dimensional hydrodynamics was used to investigate accretion disks around black holes. Since the formulation of the problem is covariant, it does not depend on the number of dimensions used. A brief summary of the equations follows.

In our work, we use the metric $g^{\mu\nu} = \mathrm{diag}(-1,1,1,1)$. Hence $\gamma = \det(g^{ij}) = 1$. We retain γ in the equations, however, because this provides us with flexibility if we choose to perform calculations in a reduced geometry such as spherical or cylindrical coordinates. For such calculations, we merely have to supply a new γ. Even in this case, however, we will take γ to be time-independent. In the following, we denote the relativistic four-velocity by u^{μ}, the baryon number density by ρ, the pressure by P, the energy density by $\rho+\varepsilon$, and the inertial density by $\rho_I=\rho+\varepsilon+P$. We use as units throughout this paper, c=1, the nuclear mass 1, and all lengths are in fermis.

Hydrodynamic equations arise from two considerations, the conservation of mass, and the conservation of energy momentum. If the mass flux is

written as ρu^μ, then the continuity equation is

$$\frac{1}{\sqrt{\gamma}} \partial_\mu (\rho u^\mu \sqrt{\gamma}) = 0. \tag{2.1}$$

Considering the 4-vector

$$V^\mu = (1, \vec{v}), \tag{2.2}$$

where $\vec{v}$ is the 3-velocity, the 4-velocity can be written

$$u^\mu = W V^\mu, \tag{2.3}$$

where W is the relativistic factor $W = (1-\vec{v}^2)^{-1/2}$. Letting $D = \rho W$, Eq. (2.1) then becomes

$$\partial_t D + \frac{1}{\sqrt{\gamma}} \nabla \cdot (D\vec{v} \sqrt{\gamma}) = 0. \tag{2.4}$$

Energy-momentum conservation arises from the divergence of the stress-energy tensor. For a perfect fluid this tensor has the form

$$T^{\mu\nu} = (\rho+\varepsilon+P) u^\mu u^\nu + P g^{\mu\nu}. \tag{2.5}$$

Rather than using the divergence of energy-momentum tensor directly, we can project it along the 4-velocity. This projection yields, after using $u_\mu u^\mu = -1$,

$$\frac{1}{\sqrt{\gamma}} u_\nu \partial_\mu (T^{\mu\nu} \sqrt{\gamma}) = - \frac{1}{\sqrt{\gamma}} \partial_\mu (\varepsilon u^\mu \sqrt{\gamma}) - \frac{P}{\sqrt{\gamma}} \partial_\mu (u^\mu \sqrt{\gamma}) = 0. \tag{2.6}$$

Substituting $E = \varepsilon W$, this equation becomes

$$\partial_t E + \frac{1}{\sqrt{\gamma}} \nabla \cdot (E\vec{v} \sqrt{\gamma}) + P\partial_t W + \frac{P}{\sqrt{\gamma}} \nabla \cdot (W\vec{v} \sqrt{\gamma}) = 0. \tag{2.7}$$

Note that our proper energy density $E \neq T^{00}$ in contrast with, for example, Ref. 5.

The momentum equation is found by evaluating the divergence of the space-like part of the stress-energy tensor,

$$\nabla_\mu \ (\rho+\varepsilon+P)\ u^\mu u_j + P g^\mu{}_j \ = \ 0. \qquad (2.8)$$

Defining a momentum $S^\mu = (\rho+\varepsilon+P)\ W u^\mu$, Eq. (2.8) becomes

$$\partial_t S_j + \frac{\partial_i}{\sqrt{\gamma}}(S_j V^i \sqrt{\gamma}) + \partial_j P + \frac{1}{2}(\partial_j g^{\mu\nu})\frac{S_\mu S_\nu}{S^t} = 0. \qquad (2.9)$$

Eqs (2.4), (2.7) and (2.9) yield five equations that we solve for the collective flow. At present, we do not include a Coulomb field or a restraining potential, due to the difficulty and cost of overlaying a three-dimensional Poisson solver. These equations form the basis only of the hydrodynamic part of the model; the details of the pion and freeze-out models will be presented in later papers.

We have chosen to employ Eulerian methods to solve the equations, due to the high dimensionality of the problem. In an Eulerian calculation, the computational mesh is fixed in space, and the fluid may be thought of as flowing through it. This simplifies the problem in that coordinates are always fixed in space, but leads to certain numerical difficulties in correctly evaluating fluxes between cells. Lagrangian methods, where comoving coordinates are used, are simpler since the coordinates flow with the fluid.[10] However, in more than one dimension, such methods lead to severe distortions of the mesh which results in the computation becoming significantly less accurate. Adaptive mesh methods[11] in which the problem forces grid points to move to regions of large gradient, suffer the same problems as do the Lagrangian formulations. Hence, the Eulerian choice is the most reasonable one for our problem.

The most difficult phenomena for any hydrodynamic code to handle are shocks. These are inherently nonlinear and thus require special care. The physics we are considering, i.e., matter impacting at high velocity,

means shocks are certainly possible. Models designed specifically to handle shocks have been investigated in one and two dimensions. Their results indicate, among other things, that shocks may be responsible for most of the entropy generation in the collisions we are considering.[12] The handling of shocks is thus a major factor in our choice of numerical method and plays a large part in the testing of the code.

Another factor in designing a hydrodynamic code is choosing between time implicit and explicit methods. In an explicit method, values at one time step are calculated from those at the previous time. In implicit methods, simultaneous equations are solved for all values to integrate them.[13] Explicit methods are simpler and less computationally demanding. Since our problem is extremely dynamic, the computational complexity of implicit methods means that they would be too expensive to use. We have thus chosen an explicit method.

The time step in explicit methods is limited by the Courant-Freidrichs-Lewy (CFL) condition. For linear equations and straightforward methods, it is easy to demonstrate that the CFL condition is required for a numerical solution to be stable.[10] This condition essentially reduces to the requirement that the time step be small enough that no signal can propagate across more than one cell width in a time step, i.e., this is a causality condition. Note, however, that this provides an upper limit on the time step only for stability of the numerical scheme.

The choice of an explicit method leads to one of our greatest sources of inaccuracy. Since the equations are not solved simultaneously as in an implicit method, there is a self-consistency to the solution which is missing. This arises especially in terms involving $W = (1-v^2)^{-1/2}$ which becomes very nonlinear as $v \to 1$. Thus, we find that the applicability of our model is limited in W. Methods have been developed for relativistic fluid dynamics which may be able to get around this problem,[11] but, as above, have been demonstrated only in one spatial dimension.

Before detailing the workings of the code in Section III, we end this section with a brief description of some of the numerical procedures common to much of the program. First, we discuss our choice of grid. We set up two interwoven, three-dimensional meshes of points. These are the cell centers and edges (see Fig. 1). The edges are denoted by subscript a, and cell centers by subscript b. This is a somewhat different notation from what other workers have used, where cell edges are denoted by index j and cell centers by j+1/2. Associated with each cell is a width, e.g., $(\Delta X_b)_j$ is the width of cell j in the X direction. These intervals carry the subscript of the point about which they are centered, rather than the grid whose differencing yields them. Thus $\Delta X_{ai} = X_{bi} - X_{b(i-1)}$. Note that this is not quite the scheme used in Ref. 8. We further impose reflection symmetry through the Z=0 plane and hence only calculate the behavior of the fluid lying above this plane.

All scalars are located in cell centers. In order to have second-order accuracy in space, vectors must be staggered with respect to the scalars. Hence, each component of a vector is centered on the face of the cell corresponding to its direction, e.g., V_x is in the midpoint of the cell with respect to the y and z directions, but at the edge of the cell in the x direction. We place the physical edges of the grid one cell in from the edge. Thus there is always one more physical location for vector values in each direction than there is for scalars.

We evaluate differential equations by operator splitting; that is, we split them into terms such as acceleration, advection, and pressure work, and calculate each such term individually. We thus reach intermediate results for each equation as we calculate terms using the values so far calculated (not just the value from the previous time step). However, as will be seen below, the velocity is calculated only once for each time step. This is because our vectors and scalars are staggered in time as well as in space and, hence, we cannot use the incomplete results from an intermediate part of the calculation that would mix quantities at different times. Once again, this can lead to problems with extreme relativistic flow due to the lack of self-consistency.

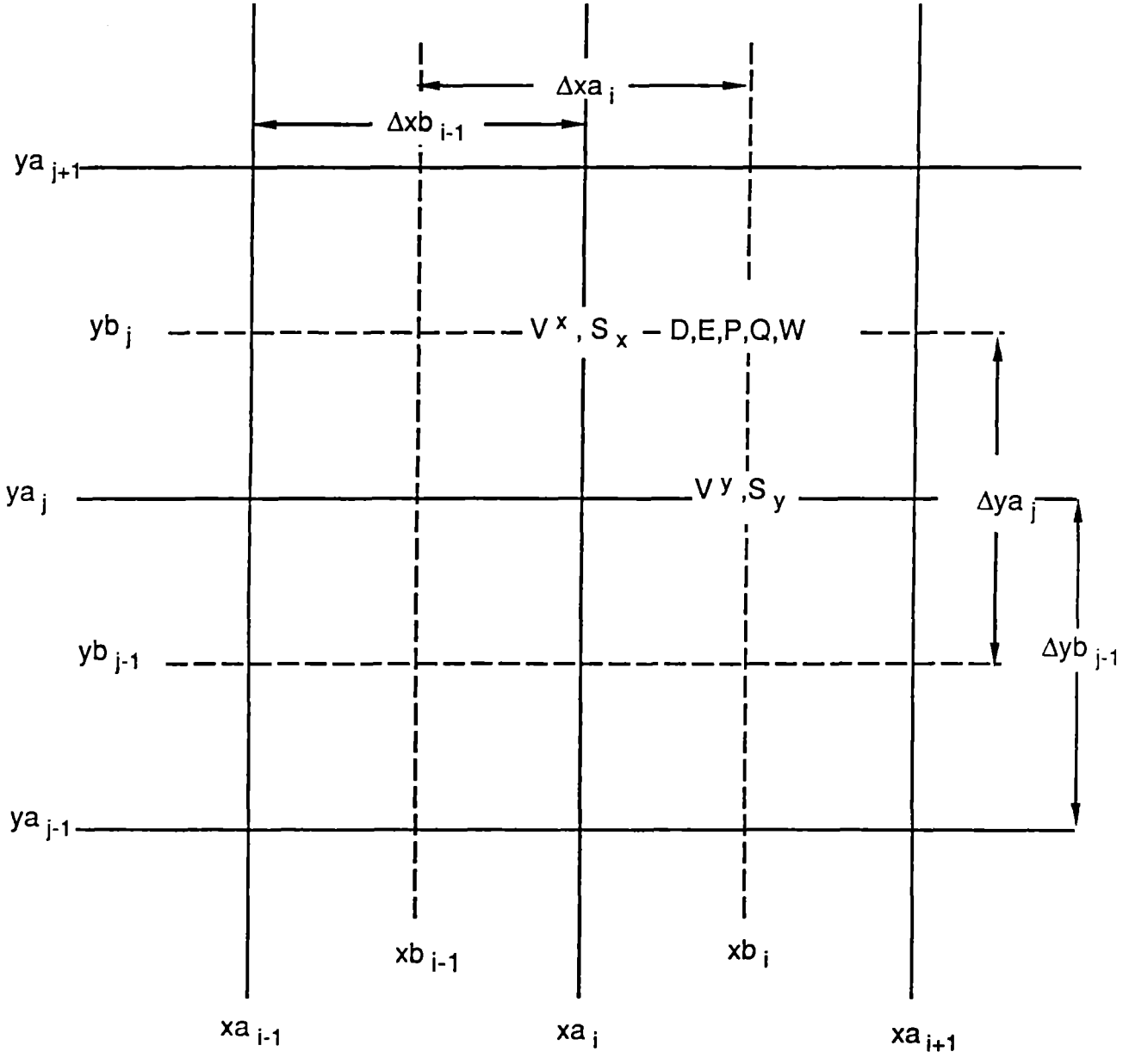


Figure 1. A typical Z_b plane is shown. V^z and S_z are located on Z_a planes at the intersection of Yb and Xb lines. Note that the intervals are labeled by the point about which they are centered, rather than the grid which is differenced to yield their value.

As stated above, advection is the most critical part of an Eulerian calculation. Numerical studies have shown[9] that relatively minor changes in the choice of an advection scheme can lead to dramatic changes in results, at least for shock tests. Hence, our choice of advection scheme was dictated by considerations of accuracy and stability. The simplest sort of advection scheme, donor cell upwind differencing[14], is only first-order accurate and may be inaccurate in regions of large gradients.

Since we wanted second-order accuracy in all calculations, we were forced to choose a monotonic scheme. These are so named since values which are interpolated are guaranteed to be monotonic if the interpolation points are. This property is important for two related reasons. In the advection term of Eq. (2.4), for instance, the spatial derivatives are evaluated in our code by taking a difference of the fluxes at the edges of the cell. The density, however, is located at the center of the cell. Hence, we must interpolate its value to an edge to evaluate the flux. Physically, there cannot be more flux coming from a cell than is available inside it at that time step. The monotonicity condition ensures this numerically. There are, however, many ways of performing this interpolation linearly, all of which yield slightly different results for shock tests.[9] Not only that, but this interpolation can be done quadratically[15] or by assuming each cell is a Riemann shock tube.[16] Due to our experience with linear interpolation, and the known robustness of the method, we have chosen to use a linear, second-order, monotonic scheme.

In our implementation of this scheme, used for both the vector and scalar flux calculations, the minimum slope between the value in a cell and those in its nearest neighbors is found. This slope is given the sign of the difference between values linearly interpolated to the cell edges. The flux is then evaluated by interpolating upstream with respect to the velocity a distance 1/2 Vδt deep from the cell edge using this slope (Fig. 2). The fluxes at either side of the cell are then differenced to yield a divergence.

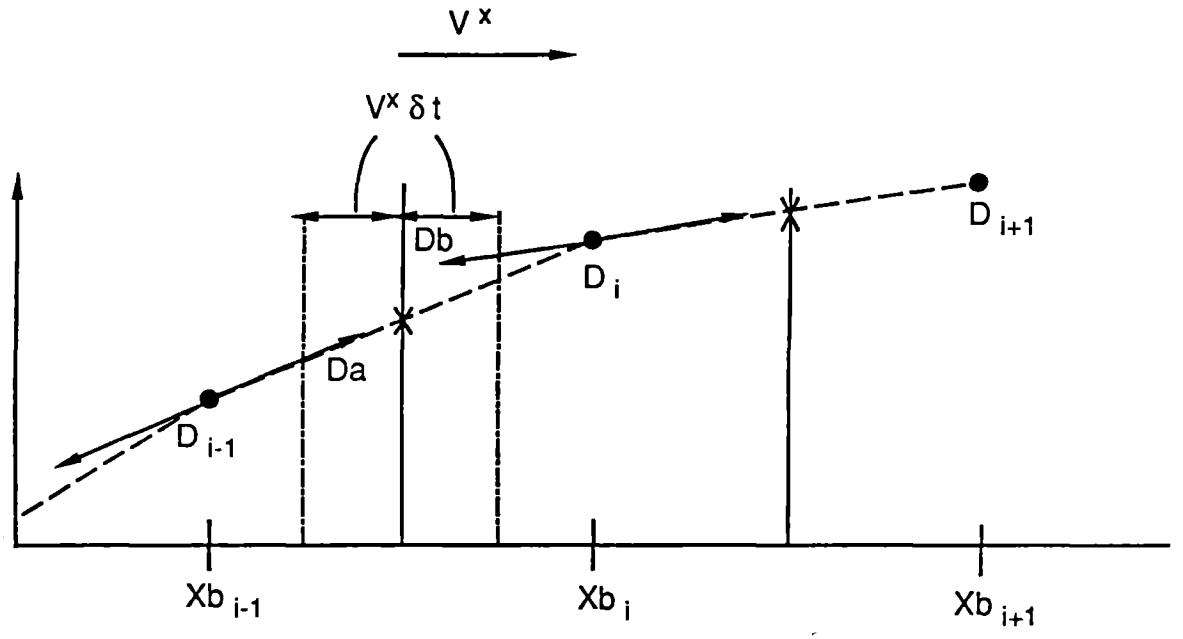


Figure 2. The advection scheme used as applied to the density flux in the x-direction. Since the difference between D_i and D_{i+1} is smaller than that between D_i and D_{i-1}, it is used for the slope. The crosses denote the interpolated values. Since V^x in the example is to the right, Da is used as the extrapolated value at the cell boundary.

All such second-order schemes require an additional numerical term, known as artificial viscosity, to stabilize them.[14] This artificial viscosity serves to reduce the dispersiveness inherent in these methods. A discussion of how such terms arise may be found in any standard reference on the numerical solution of hyperbolic partial differential equations[14] and will not be dealt with here. We should note that we have included a linear term,[17] which will be described in the next section, as well as a quadratic one. We have implemented them in such a way that only heating is possible from the viscosity, even though it is not strictly positive.

In one-dimensional calculations, the artificial viscosity originally was given by

$$Q \sim (\Delta V)^2 D_I, \qquad (2.10)$$

where $D_I = W\rho_I$, and $\Delta V = V_{j+1} - V_j$ for $\Delta V<0$ and 0 otherwise, i.e., it is in effect only during compression. Its effect is to spread out shocks over a finite number of zones, and thus the interiors of shocks are not handled correctly. Since we are not interested in the actual structure of shocks, but their effect on the nucleus, this is not a concern. Equation (2.10) must be generalized to three dimensions

$$Q \sim D_I \Sigma \Delta V_i^2. \tag{2.11}$$

for it to be useful in our code and, in addition, the compression condition is checked for each direction individually, rather than for the difference as a whole. The precise form of the artificial viscosity that we employ can be derived nonrelativistically from the Hugoniot shock conditions.[18] By eliminating the postshock density and using a gamma-law equation of state, one finds that

$$P_2 = P_1 + \frac{\gamma+1}{4} (\Delta V)^2 \rho_1 + \rho_1 |\Delta V| \left(\left(\frac{\gamma+1}{4}\right)^2 (\Delta V)^2 + C_s^2 \right)^{1/2}, \tag{2.12}$$

where the subscripts 1,2 refer to pre- and post-shock conditions, respectively, $\Delta V=V_2-V_1$ and C_s is the preshock sound speed. By considering the limits $\Delta V >> C_s$ and $\Delta V << C_s$, we arrive at the form

$$Q = \frac{\gamma+1}{2} (\Delta V)^2 \rho_1 + C_s |\Delta V| \rho_1 \tag{2.13}$$

for the artificial viscosity, which should insure preservation of Hugoniot relations across the shock. This is true for the nonrelativistic case but is not appropriate relativistically. The pressure jump for a relativistic shock, as given by the relativistic Hugoniot relations, suggests that $\Delta(S/D)$ rather than ΔV is the appropriate difference to use. While we have not been able to derive an equation analogous to (2.12) for the relativistic case, the result just mentioned, along with dimensional considerations, has led us to use a form similar to (2.13);

$$Q = \frac{\gamma+1}{2} (\Delta(S/D))^2 D^2/W^2 \rho_I - C_s D \Delta(S/D) \qquad (2.14)$$

We use monotonically centered differences in calculating $\Delta(S/D)$, as opposed to the usual edge differences.[19] This has been shown to yield extremely narrow and stable shocks in our tests.

The artificial viscosity enters generally as a correction to the pressure. In a completely consistent covariant formulation, artificial viscosity would thus enter the equations everywhere the pressure does. Two problems arise from this. The first is another example of the lack of self-consistency inherent in explicit codes. Q and W are nonlinear functions of each other and of the other hydrodynamic variables and, thus, it is difficult to properly time center terms involving them. As noted above, this was a problem when we were considering W alone. With the introduction of Q, the problem has worsened. This problem can be attacked to some degree by iterating approximations to certain terms in the operator splitting.

The other problem may be more fundamental. In Ref. 8 it was found that introducing the artificial viscosity everywhere the pressure appears did not yield stable numerical solutions to the hydrodynamic equations. It has been known for many years[20] that first-order formulations of the relativistic Navier-Stokes equations[21] yield unstable equilibria. Numerical solutions have been found to relativistic hydrodynamic models with viscosity,[22] but neither of these models handle the viscosity in a consistent fashion. In both instances, the viscosity is calculated as a perturbation to the ideal flow by calculating it from the previous rather than the current timestep. Such a procedure would not be expected to apply near large gradients, as are found near a shock. Following Ref. 8, we have included the artificial viscosity in the heating term, i.e., the last term in Eq. (2.7), and the acceleration term, the spatial derivative of P in Eq. (2.9). Since the form of the artificial viscosity (2.14) yields a small value in all but a very few cells, we do not include Q in any of the other places P appears.

The issue of multidimensional artificial viscosities has not been treated widely in the literature.[18] In the terms where we employ the artificial viscosity, heating and acceleration, we need nonscalar values. The purpose of the artificial viscosity is to spread out the shock parallel to its direction of propagation and, hence, it makes sense for this quantity to be higher order, e.g., a tensor, in order that it have a direction. In our calculations, we perform our artificial viscosity calculations in each direction separately, yielding the diagonal components of an artificial viscosity "tensor", Q_{jj}. Then the total resulting acceleration is of the form

$$\dot{S}_i = \sum_j \partial_j (Q_{ij} + P) \qquad (2.15)$$

and the heating term $\sum_i (P + Q_{ii})\ \partial_i WV_i$.

The final numerical issue to be discussed is boundary conditions. For symmetric collisions, at least, we can use reflection boundary conditions at 0 impact parameter, and a modified reflection condition for finite impact parameters. The reflection condition is that scalars and tangential components of vectors are constant across the edge of the physical region of our grid and normal vector quantities are set to 0 at the edge. The modified reflection conditions are that quantities are reflected about the y axis as well as the x, as shown in Fig. 3. At the other extreme, those boundaries of the physical region that are not reflecting should be transparent. We insure this by only allowing outgoing flux at the boundaries and by setting the exterior scalar quantities to a linear extrapolation of their neighboring interior zones so that their outflow will not be impeded.

This concludes the general discussion of the most important numerical issues faced in the code. In the following section, we will detail the specific workings of the code itself.

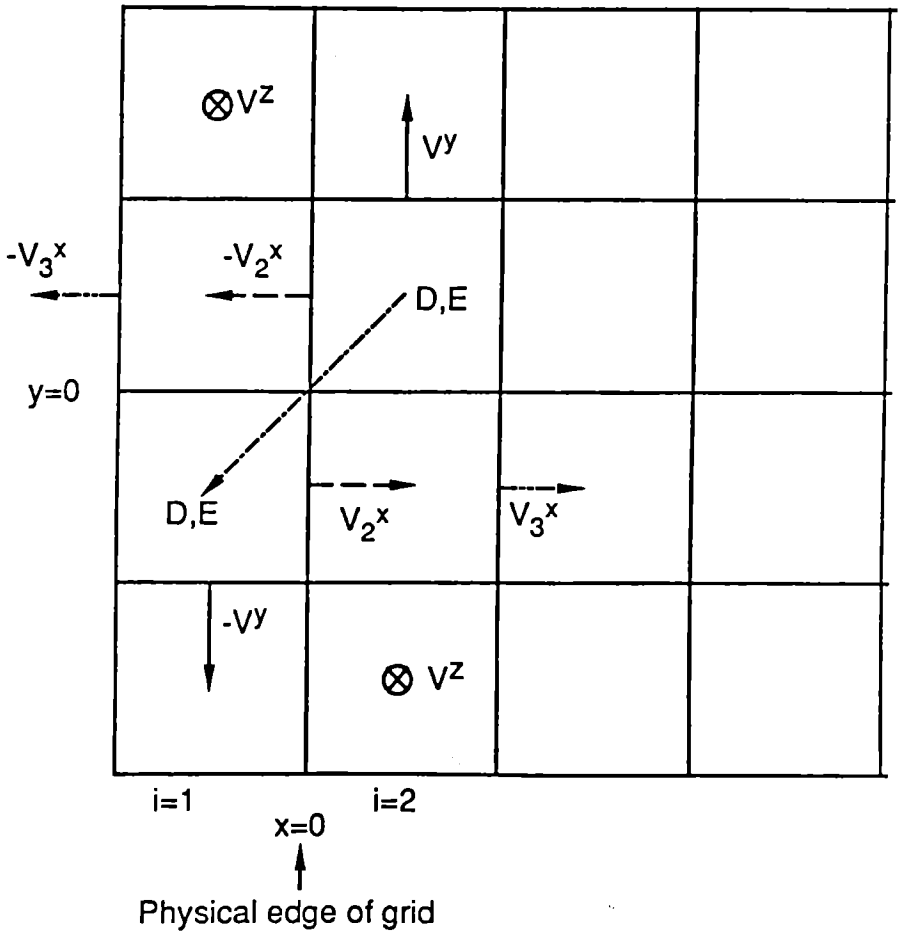


Figure 3. Reflection-inversion boundary conditions. i=2 values are imposed at i=1 according to this scheme. Magnitudes are always preserved, but the directions of V^y and V^x are inverted. Note that V_2^x (dot-dash line) is inverted to i=2, while V_3^x (double-dot dash line) is inverted to i=1.

SECTION III

The program consists of a cycle of subroutines used to perform the operator splitting algorithm mentioned above. The integration is divided into acceleration terms, advection terms (including pressure work), terms from the time integration of W, and those from the artificial viscosity. In addition, the code has a generator for initial conditions, and graphics and printout routines. The general scheme of the program is seen in Fig. 4. Details of how to run the program are found in Appendix I.

Even with sharing of all matrices between subroutines through the use of common blocks, large grids tend to use a large fraction of available memory when the program is compiled with all matrices stored internally. As a consequence of having CRAY X-MPs available with their large, solid-state disk (SSD) memory, we have designed the code to run with a fast external memory. Even though the SSD is much faster than an ordinary disk, we have tried to minimize the number of times it must be accessed and have only up to three x-y planes of values available at any time. Whenever possible, we read and write to the SSD in these sets of three but, whenever we difference or recenter in z, the planes must be handled individually. By cycling the indices of the three planes modulo 3, once three planes are read in, new planes can be read in and easily followed.

We maintain three types of organization of the memory. The scalars D, E, P, W, γ, and Q, as well as a scratch vector A1 are serially organized and separate. Thus, each has its own file in the SSD and is organized by planes. These values are read and written in single z-planes by the routines SREAD1 and SWRIT1, respectively, and in sets of three planes by SREAD3 and SWRIT3. The vectors S and V are organized somewhat differently. We assume that all components of these 3-vectors (note that neither is a 4-vector) will be needed when any one is and, hence, they are organized by component for each plane. Thus, there will be a VX plane followed by the corresponding VY and VZ planes, and then

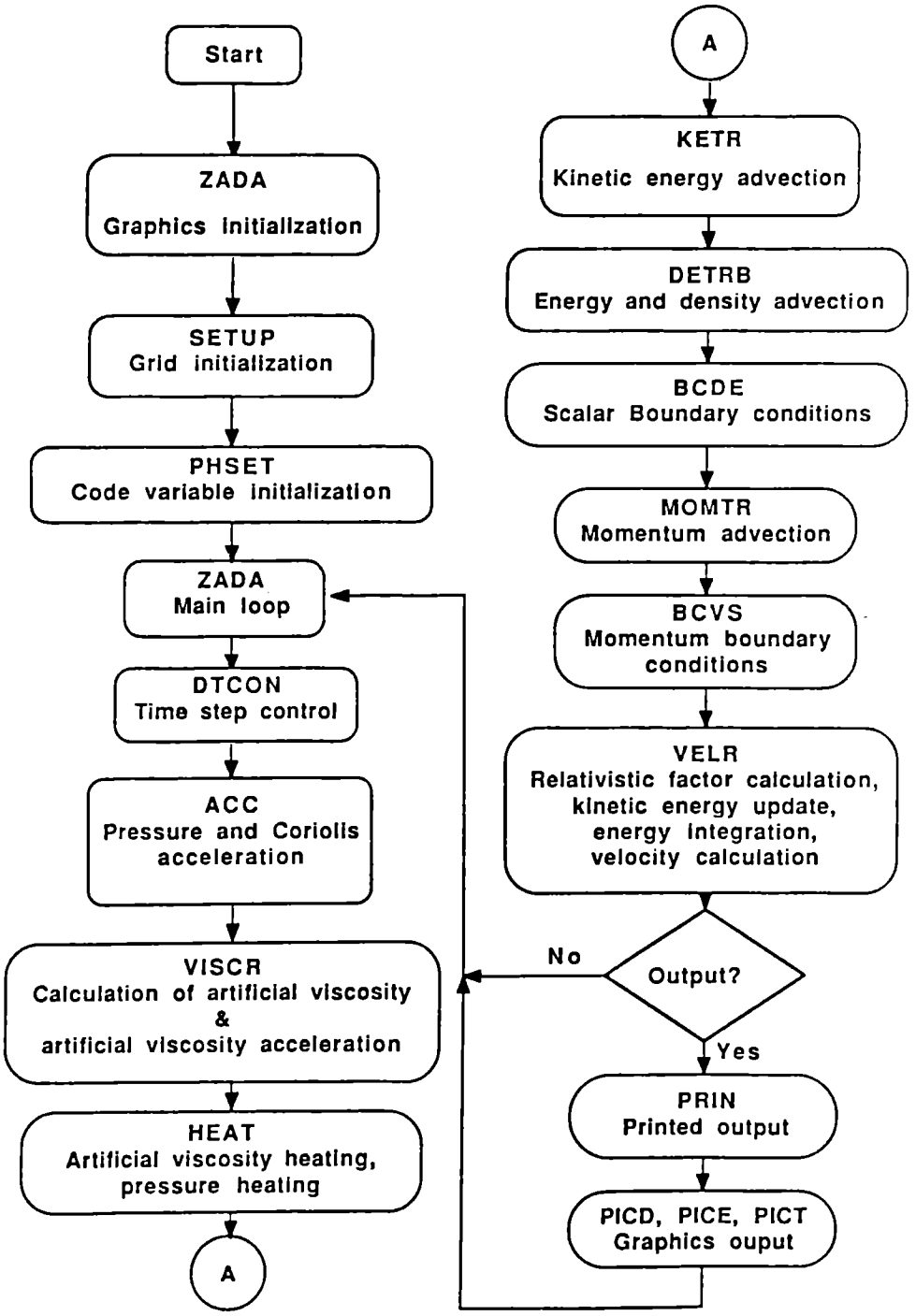


Figure 4. Flow chart for the program. Note that the storage management routines are not shown, as they are used in nearly every routine.

the next VX plane. These values are read and written by VREAD1, VREAD3, VWRIT1 and VWRIT3. Finally, we have a scratch file with the capacity for four full grids. This is accessed by WREAD1, WREAD3, WWRIT1 and WWRIT3.

The initialization routine, SETUP, is called from the main routine ZADA, as are almost all other routines. Values are read into ZADA in namelist format into the array SET. Table I lists the assignment of parameters. The program can be restarted from dump files and parameters respecified in the same way. When the parameters are read in, the routine SETUP sets up the grids. The generator routine, PHSET, then initializes the thermodynamic variables. Even though the nuclear skin is only 1 fm across, a spherical shell of this thickness and uniform density for a ^{208}Pb nucleus, for example, would contain almost 40% of the mass. Hence, we feel the surface profile is important. We thus require the nucleus to have a realistic profile and, hence, scaling solutions[23] are not appropriate. This requires finer gridding so that we may approximate the steep gradient in the skin region accurately. Once the density profile is set, we use the generator routine PHSET to derive the other code quantities from the equation of state, the initial velocities, and the definitions of D, E, Q, and S above.

The timestep is calculated in routine DTCON. Three CFL-like criteria are applied. The first is the ordinary CFL condition. The local sound speed is estimated from the thermodynamic formula:

$$c_s^2 = \frac{\rho dP/d\rho|_S}{\rho_I} . \qquad (3.1)$$

The inverse time corresponding to this is given by dividing this speed by the smallest dimension of the cell. The second characteristic inverse time is related to the artificial viscosity and is given by $c_{qi}=|\Delta V_i|/\Delta X_i$ if $\Delta V_i<0$. Finally, we can let our grid move as well, and thus there is a characteristic inverse time $c_{gi}=|V_i-V_{grid}|/\Delta X_i$. Rather than choosing the minimum

Table I. Input parameters for the code.
Default values are not supplied by the program.

```
c      set(1)  = km
c      set(2)  = jm
c      set(3)  = im
c      set(4)  = symmetry parameter
c              = 1  z=0, k=2 reflection
c              = 2  z=0, k=2 and y(j=2) reflection
c              = 3  z=0, k=2 and x(i=2) reflection
c              = 4  z=0 and y(j=2) and x(i=2) reflection
c              = 5  z=0, k=2 reflection, for x(i=2), reflect x, invert y
c      set(5)  = maximum value for density contours ( 20 contours )
c      set(7)  = switch for contour smoothing ( 0 = smoothing off, 1 = on )
c      set(8)  = time constant (fm/c) for print ( should be an integer fraction
c                of set(21), the picture cycle )
c      set(9)  = time constant (fm/c) for dumps
c      set(10) = problem time (fm/c)
c      set(11) = maximum temperature (MeV) for temperature contours (20 contours)
c      set(12) = time control
c      set(13) = viscosity coefficient
c      set(14) = threshhold in q/p for kinetic energy update
c      set(15) = dx
c      set(16) = dy
c      set(17) = dz
c      set(18) = dx ratio for exponential zoning
c      set(19) = dy ratio for exponential zoning
c      set(20) = dz ratio for exponential zoning
c      set(21) = time constant (fm/c) for pictures ( should be an integer
c                multiple of set(8), the print cycle )

c                        *** nuclear calculation parameters
c      set(22) = eos switch
c                1 -- gamma-law ( requires input gamma )
c                2 -- quadratic ( requires input K )
c                3 -- BCK       ( requires input gamma and K )
c                4 -- Sierk-Nix ( requires input K )
c                5 -- Skyrme    ( requires input a, b and nu )
c      set(23) = gamma or "a (MeV)" for EOS
c      set(24) = nuclear incompressibility coefficient K or "b" (MeV)
c      set(25) = nu, exponent in Skyrme EOS
c      set(27) = b/2 (b = impact parameter, fm)
c      set(28) = offset of nuclei from the origin in the x-direction (fm)
c      set(29) = mass number
c      set(30) = nuclear charge
c      set(37) = relativistic gamma-factor
c      set(40) = multiplies velocity to determine direction (i.e., +1 or -1)
```

characteristic time, we maximize these quantities over the mesh, and then take the timestep proportional to the inverse of their sum. The timestep is reduced for early steps to let initial conditions smooth out.

The initial part of the main program loop performs the Lagrangian parts of the calculation, the acceleration and heating terms. The pressure acceleration is performed in routine ACC and is a straightforward differencing of the pressure but not of the artificial viscosity. For calculations in a rotating frame, the centrifugal acceleration is also calculated here.

The next routine is VISCR, in which the artificial viscosity is calculated. Both linear and quadratic terms are calculated for each direction, as described above. The resulting acceleration is calculated in this routine, and the tensor diagonal components as well as their sum are saved for later use.

The pressure and artificial viscosity heating terms are calculated in routine HEAT to finish the Lagrangian part of the time step. The advection of W is calculated according to the monotonic scheme described above. Since W is a face-centered scalar, the differencing in each direction is the same, unlike for the momentum as will be shown below. Due to the use of the SSD, however, the terms look slightly different for advection in the z-direction. The artificial viscosity heating is calculated first. Since it is difficult to time center Q properly,[9] we do a simple forward time differencing on the energy density, and do not try to recenter it. The pressure heating can be time centered more easily, however, and the term we calculate is

$$E^{n+1} - E^{n} = -\delta t \left(\frac{p^{n} + p^{n'}}{2}\right) (\partial_i W V^i)^{n} \qquad (3.2)$$

where

$$p^{n'} = p + \frac{\partial p}{\partial \rho}\Big|_{s} \delta\rho$$

and $\delta\rho$ is the variation in ρ due to the term we are considering. For the heating, using the thermodynamic relationship $d\varepsilon=-PdV=P/\rho^2 d\rho$

$$\delta\rho/\rho = -(\nabla\cdot VW\gamma)/(W\gamma). \qquad (3.3)$$

After the energy is updated from the artificial viscosity heating, the pressure is recalculated in routine EOS on a plane-by-plane basis. This routine calculates the pressure given the velocity and energy and baryon densities of the matter. There are generally several equations of state available, which are chosen with an input parameter. This flexibility of choice of equation of state is important since not a great deal is known of this aspect of the physics. One of the primary motivations for this model is a greater understanding of how the equation of state affects experimental observables in heavy ion collisions.

Before performing any advection, we want to predict what the kinetic energy should look like after all advections in the zones where artificial viscosity heating has taken place, since these are where the largest error may have occurred. This is because where the artificial viscosity is large, $\Delta(S/D)$ is large, and hence the gradient may not be well approximated linearly. Thus, we calculate the kinetic energy density

$$KE = T^{00} - D - E$$

$$= (D+E)\,(W-1) + P(W^2-1) \qquad (3.4)$$

in KETR and advect it here as we advect all other scalars in order to provide an estimate of what it should be after all advections.

We then calculate the energy and density advection terms in DETRB as given by the monotonic scheme described above. We finish this part of the updating of the scalar terms by imposing boundary conditions in routine BCDE. DETRB is the only routine that deals with the baryon density, so

the density is handled in a conservative fashion. We have numerically verified this by summing over the fluxes as they leave the grid and integrating over the matter in the grid. The sum of these is always constant to machine accuracy.

The computational cycle continues with momentum advection in routine MOMTR. The general scheme of advection is as described above in Section II, but a few changes must be noted. First of all, due to the scheme we have developed to take advantage of the SSD, all z transport is calculated first. Second, only when a momentum component is transported parallel to itself does it need to be recentered. This is because momenta are only edge-centered with respect to the direction they point to, i.e., x-momenta are zone-centered in y and z, but edge-centered in x. Hence, there is a slight difference in the way that the three momenta are advected in any given direction.

Following the momentum advection, the boundary conditions on the momentum are imposed. In the reflection-inversion case, the boundary conditions have already been used by identifying edge cells with reflected interior cells. Nevertheless, the routine BCVS imposes the boundary condition at all boundaries for all our choices.

Routine VELR is then called to perform several functions. We first update the relativistic factor W using

$$W = (1 + \Sigma(S_i/D_I)^2)^{1/2}. \qquad (3.5)$$

With this new W, we recalculate the kinetic energy density (3.4) in cells where Q/P is greater than a threshold given by an input parameter. The difference between the kinetic energy density, as given by the advection in KETR, and this one is then added to the internal energy density E. We perform this update to approximately correct the error mentioned above. We update into the internal energy E even though other quantities are used since, for example, we must conserve baryon number. Next we carry out the

integration of $\partial_t E = -P\partial_t W$. Since there are differentials with respect to time on both sides, this equation can be integrated[8] and the term $P\partial_t W$ handled easily. A time-centering procedure for the pressure similar to the one used in the heating terms is applied here as well. In this case, we derive $\delta\rho$ from the condition that the lab density is not changed by this update, although a new W has been calculated. Denoting updated values by a prime, D=D' implies that

$$\frac{\rho'}{\rho} = \frac{W}{W'} \tag{3.6}$$

and hence

$$\delta\rho = -\rho\left(\frac{W'-W}{W'}\right).$$

After the energy is updated, the pressure is again calculated in routine EOS.

This subroutine's last function is to calculate the velocity to be used in the rest of the program, especially for advection. The velocity is

$$V_i = S_i/(W\, D_I) + \Omega_i. \tag{3.7}$$

Here $\Omega = \omega \cdot x$ is the rotation vector if the frame is rotating with angular velocity ω. Thus V_i is the velocity in the <u>rotating</u> frame as viewed from the center. The momentum, however, is in the <u>lab</u> frame. Boundary conditions on V are also imposed in this routine.

SECTION IV

In developing a model such as ours, it is crucial that the numerical procedures do not introduce spurious effects into the problem. We have thus carefully tested the program against known analytic solutions to the relativistic hydrodynamic equations. This gauges the accuracy the code is capable of delivering as well as explores its limitations. Due to the complicated initial conditions and interactions of the real problem we

are trying to solve, we cannot solve it analytically; we feel, however, it is best to test the numerical aspects of the model as extensively as possible in order to understand the results.

Analytic solutions to the hydrodynamic equations test the program as a whole, but it is more efficient to test as many parts of the code as possible separately to identify potential problems. Since we have implemented the equations in an operator-splitting form, various physical processes are isolated naturally into the different subroutines and, hence, the testing is both easily done and easily interpreted. As will be noted later with the analytic tests, these trials are generally performed in each of the spatial directions separately since they are intrinsically one- rather than three-dimensional.

It is possible to rewrite the hydrodynamic equations in terms of any of the thermodynamic variables and, hence, they can be interpreted as entropy flux transport equations as well. In the absence of a source (i.e., a shock), entropy is merely transported through the grid and all calculations should lie on adiabats. We can achieve such a condition by setting the coefficient in front of the artificial viscosity to zero, in which case all calculations should be adiabatic. Given an ideal gas equation of state we should find that, for example, under compression the fluid follows the behavior $P=\rho^{\gamma}$. We have performed this test on the code by, as mentioned, setting the artificial viscosity to zero and choosing such a small timestep that very little matter is transported in or out of a zone during a test. In addition, we must choose initial conditions so as to ensure that compression does occur. We do this by choosing the velocity such that the fluid is at rest at the edges and moving with its maximum velocity in the center, with the velocity linearly interpolated to the points in between. At every cycle, the velocity is set to this initial condition. This results in uniform compression to one side of the midpoint, and uniform decompression on the other. As can be seen from Fig. 5, adiabatic compression is observed for a wide variety of velocities, including moderately relativistic ones.

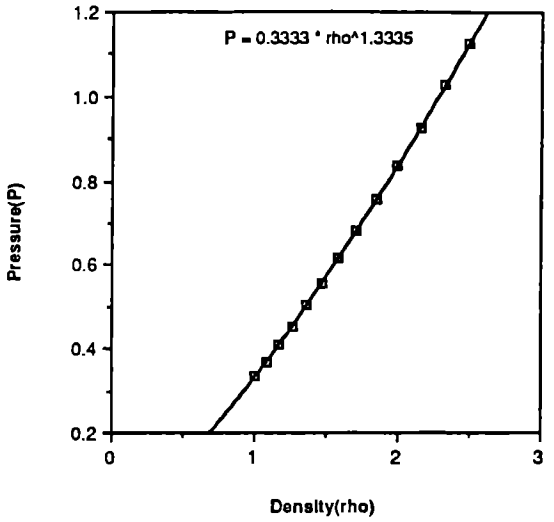


Figure 5. Pressure as a function of density for an ideal $\gamma=4/3$ gas in the first compression test of Section IV. Three different maximum velocities are shown and all are fit quite well by an exponential of the form $P=a\rho^{4/3}$.

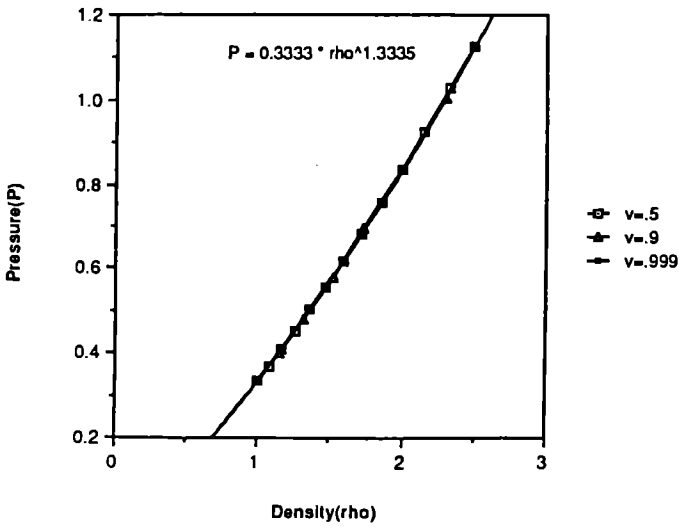


Figure 6. Pressure as a function of the inverse of the relativistic pressure to test the $E=-P\partial_t W$ term of Eq. 2.7. Since we hold $D=1$, $\rho=\frac{1}{W}$ and, hence, $P=a(\frac{1}{W})^{4/3}$ behavior is seen.

This confirms that the heating routine works correctly. The next part of the calculation that can be isolated and tested is the third term of equation (2.7), which has no nonrelativistic correspondence and is thus especially important in relativistic flows. By having a uniform velocity throughout the grid, which is increased by hand each timestep, we observe the compression in the furthest downstream zone. As with the previous test, we set the artificial viscosity coefficient to zero and chose a small enough timestep to ensure very little transport. The results of that calculation are shown in Fig. 6 and are seen to very closely follow the adiabat.

The remaining ideal adiabatic behavior that can be tested in a simple way is the transport of quantities through the mesh. While the previous calculations need only be carried out in one dimension, with this test it is possible to test all three dimensions simultaneously. We have performed two versions of this test. In both cases, we set up a realistic nucleus in the grid, in terms of the scalar variables which characterize the fluid. For the first test, we give the nucleus no initial velocity, in order to see if the way that the problem is set up results in a nucleus stable against decay. While this may seem an obvious and desirable characteristic for any ground-state description of a stable nucleus, most cascade models allowed nuclei to spontaneously evaporate particles until it was discovered how to add a confining potential to the model.[24] While we do not have a potential in our model, since the pressure is set to zero for densities less than that of normal nuclei, we should preserve our nuclear shapes for ground states. We have observed this stability over a very large number of cycles in these tests.

The other transport test is to set up a nucleus with a uniform initial velocity and observe if it propagates through the mesh without appreciable distortion. This is a general test of all ideal aspects of the calculation, and our program performs this with no significant change in energy. By running the problem until one or another edge is reached, we can also ensure that transparent boundary conditions are correctly dealt

with. We find no appreciable pileup of density or energy at the edges and, as well, when our reflection-inversion boundary conditions are used, the nucleus reappears as expected.

The final, most difficult tests exercise the non-ideal aspects of the calculation. These involve one-dimensional shock tests where analytic solution is possible. In general, these problems exhibit scaling solutions, i.e., they are functions only of z/t. This simplifies the equations considerably. The two tests we have performed are the relativistic shock tube and the relativistic wall shock.

The relativistic shock tube is the analogue of the Riemann shock tube, which is a well known test problem for classical hydrodynamic codes. The physical situation is an infinite tube filled initially with a gas on either side of a diaphragm. This gas is held at different densities or temperatures on either side of the diaphragm. At the start of the problem, the diaphragm is removed. A rarefaction wave proceeds through the higher density material, while a contact discontinuity and shock proceed through the lower density one. The relativistic version is achieved by setting up the problem so that the internal energy of one of the sides is large with respect to its mass. Then the material flowing into the low-density region will move relativistically. Analytic solutions to this problem are known,[25] and thus the results of the numerical calculations can be verified. We have performed relativistic shock tube tests with the initial conditions as in Ref. 8, and the results are shown in Fig. 7, along with the analytic results.

The final test which has been performed on the code is in some sense the most difficult, since it tests shock phenomena the most rigorously. This is the wall shock, in which cold matter hits a perfectly reflecting wall. The fluid is stopped, and a shock proceeds out from the wall, leaving highly compressed, shocked matter behind it. A three-dimensional version of this is Noh[26] problem, in which matter spherically implodes.

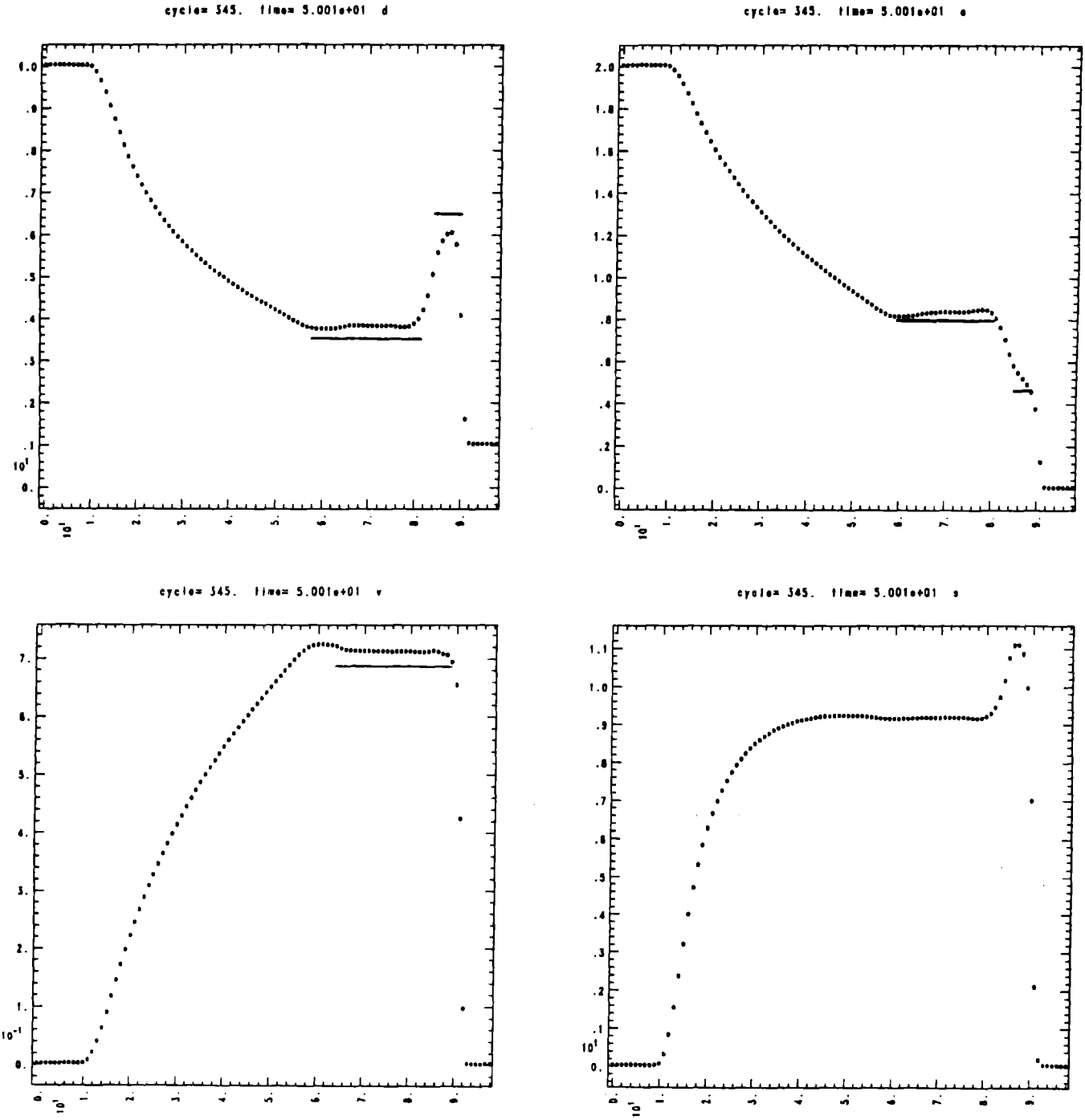


Figure 7. Relativistic shock tube results for ρ_{Left}/ρ_{Right}=10., and vanishing initial energy density to the right. Clockwise from upper left are the density, energy, momentum and velocity. Analytic values are the solid lines.

The analytic solution to this problem is given by Rankine-Hugoniot shock conditions, which can be derived for relativistic as well as nonrelativistic flows. We have performed this test for a wide range of velocities and, in fact, this test helps to define the boundaries of applicability of our model. In order to compare our code to a previously published one, we have performed the shock tests of Centrella and Wilson.[27] At values of W, ranging from near 1 to W greater than 10, we get very good results with postshock densities generally less than 10% different from the theoretical values. This is achieved without varying the artificial viscosity to an unrealistically large value as has had to be done with conventional artificial viscosities. We compare our results in Fig. 8.

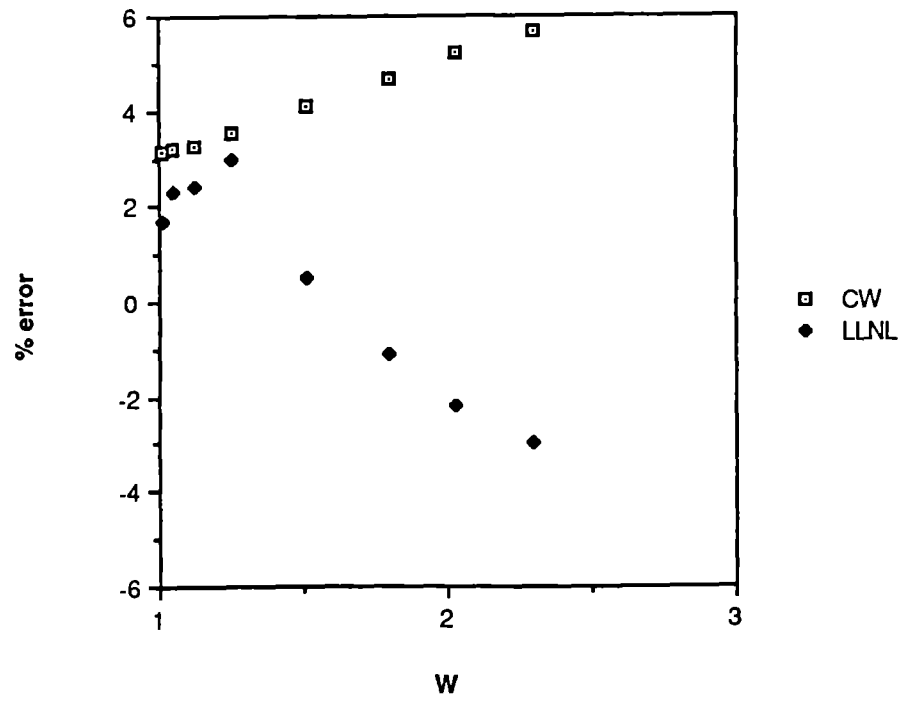


Figure 8. Percentage error in compression for the relativistic wall shock. Crosses denote the current code, squares for the results from Ref. 27. Note that the latter used an artificial viscosity constant of 7 to achieve their accuracy, while the present calculations are with parameters as used in nuclear calculations.

In conclusion then, we have devised and performed a number of tests to verify both that separate components of our calculations are proceeding correctly and that the components work together correctly as a whole. We feel that these tests provide us with a comprehensive overview of the capabilities of our model as well as to define the parameter space in which we can expect it to perform correctly.

SECTION V

We have developed a hydrodynamic model in order to simulate heavy ion collisions at high energy.[28,29] We feel that the hydrodynamic approximation is justified in these collisions due to the significance of collective effects in high energy nuclear collisions. For these collisions, the hydrodynamic model should provide a good description of the collective flow observed. To make real experimental predictions, however, we must add features to the model which will be described in forthcoming papers. The first of these is a pion model.[8] Since pions carry a large fraction of the energy and entropy of such reactions, we have modeled their production dynamically as opposed to chemically. The last piece which must be added is a light fragment freezeout. Since heavy fragments are difficult to detect, most data are on elements lighter than carbon. Hence, we must also incorporate a physical model that will take us from our bulk flow to the observed light fragments.

ACKNOWLEDGMENTS

We would like to thank Jim LeBlanc and Randy Christensen for many helpful discussions during the development of this model.

APPENDIX I. RUNNING THE COMPUTER PROGRAM

In this appendix we will present a brief guide to running the program itself. Note that the pion model is run at the same time as part of the compiled code, so running the program will involve the pion model as well, provided the input parameters are chosen appropriately.[8]

All input to the program is through an input file, which has the namelist parameters mentioned above. Table I presents a list of the SET parameters appropriate to the hydrodynamics part of the model. There are further parameters required for the pion model which are described in its manual.[8] The program has the ability to occasionally dump all working variables into files to be used for restarts, so the first part of the input file must tell the program if this is the case. A variable IRESTART is used to specify the name of a file; if IRESTART is blank, then no restart file is used, otherwise the file specified by the Hollerith string is loaded. Following IRESTART, the string "$END" must appear on a separate line to denote the end of a namelist.

The SET parameters follow, as described in Table I and Ref. 8. The hydrodyanmics code may be run alone by specifying SET(34)=0. Following these parameters, another "$END" line appears. If the run is a restarted dump file, the SET parameters from the previous run are used, and the above parameters are ignored. In this case, changed values of the SET parameters may now be specified, followed again by "$END". Even if the run is not a restart, the last "$END" must be provided.

The input file must be given the name "INxxxx", where xxxx is a string of up to four alphanumeric characters. Then the program is run by giving the controlee name, generally "xpress", followed on the same line by "xxxx" from the input file name. If not, the program is invoked with just the controlee name and will search for the file "in3d". If there is no appropriate input file found, the program will not run.

The program will generate dump and output files as specified by the input name. Three kinds of files are generated: dumps, graphic output, and text output. The dumps are generated at intervals specified by SET(9). Each is given the name "xxxxnnnn", where nnnn is the time of the dump in tenths of a fm/c and is a zero-filled, four-digit number. Graphic output is in DLTV format and is found in a family of files with the names "d0xxxx00" through "d0xxxxnx". If "xxxx" is less than four characters, they will be padded on the left with random alphabetic characters so that the graphics filenames are always eight characters. The main text output is in the file "outxxxx" and extra output, generally for debugging or special analysis is found in "outdxxxx". Information from the pion model is given in "pionxxxx".

APPENDIX II. OUTPUT FOR THE PROGRAM

Output from the program is found in three types of files, as mentioned in Appendix I. The first, dumpfiles, are used only to restart the program and, hence, will not be described here. The other output files will be described below. Note that these descriptions are current as of late 1987 and are subject to change depending on the development of the model.

Due to the large number of cells in a three-deimensional calculation, cell-by-cell information in text form is of limited usefulness. Thus, the text file "outxxxx" contains cell information only for cells along the three axes. For each cell along the axis, its proper density ρ (in units of normal nuclear density), proper energy, three-velocity and momentum along the axis and pressure are given. In addition, at each time of printing, the total density in the grid is given, as well as the integrals of kinetic and internal energies. The entropy, density-weighted average temperature and maximum density are printed. Finally, the amount of material and energy that has flowed out of the grid also is shown.

Graphical output is generally much easier to grasp in a calculation such as the code performs and, hence, we provide a number of plots at intervals specified by SET(21). There are contour plots of the proper energy and density and temperature in the plane Z=0, as well as density profiles along the x- and y-axes. We also provide color plots of the density in this plane, with pions projected from all Z-values. The length of the arrows indicate the pion momenta, and their tails lie at the present projected position. These color plots cannot be viewed on a TMDS; they can, however, be previewed on a MAGIC terminal. The color table may be specified by substituting the routine RGBGEN in the program. Following Danielewicz and Odyniec,[30] we also plot transverse momentum versus rapidity.

REFERENCES

1. M.R. Anastasio, L.S. Celenza, W.S. Pong, and C.M. Shakin, Phys. Rep. 100, 327 (1983).
2. J. Boguta and R. Bodmer, Nuclear Physics A292, 413 (1977); J. Boguta and H. Stöcker, Phys. Lett. 120B, 289 (1983); R. Brockmann and R. Machleidt, Phys. Lett. 149B, 149 (1984).
3. S. Nagamiya and M. Gyulassy, "High Energy Nuclear Collisions", in Advances in Nuclear Physics 13, 201 (1984), edited by J. Negele.
4. J. Bondorf and J. Zimanyi, Phys. Scr. 24, 758 (1981).
5. For reviews, see H. Stöcker and W. Greiner, Phys. Rep. 137, 758 (1986); R.B. Clare and D. Strottman, Phys. Rep. 141, 177 (1986).
6. H. Kruse, B. Jacek, J.J. Molitoris, G.D. Westfall, and H. Stöcker, Phys. Rev. C31, 1770 (1985); J.J. Molitoris, A. Bonasera, B.L. Winer, and H. Stöcker, Muhlenberg College, MUHPHY 01 (1986).
7. A.A. Amsden, F.H. Harlow, and J.R. Nix, Phys. Rev. C15, 2015 (1977).
8. T.L. McAbee, J.R. Wilson, P. D'Alessandris, J.A. Zingman, C.T. Alonso, LLNL Internal Report UCID-21125.
9. J. Hawley, L. Smarr, and J. Wilson, Ap. J. 277, 296 (1984), and Ap. J. Supp. 55, 211 (1984).
10. See, for example, D. Potter, "Computational Physics", (1973, John Wiley and Sons) Chapter 9.
11. M. Norman and K-H Winkler, "Why Ultrarelativistic Numerical Hydrodynamics is so Difficult", in K-H Winkler and M. Norman, "Astrophysical Radiation Hydrodynamics", NATO ASI Series C188 (1986, D. Reidel).
12. W. Scheid, H. Mueller, and W. Greiner, Phys. Rev. Lett. 32, 741 (1974).
13. P.J. Roache, "Computational Fluid Dynamics" 230, (1982, Hermosa Publishers).
14. R.D. Richtmeyer and K.W. Morton, "Difference Methods for Initial Value Problems", Chapter 12 (1967, John Wiley).
15. P.R. Woodward, "Piecewise Parabolic Methods for Astrophysical Fluid Dynamics", UCRL-90009 (1983).

16. J-P Blaizot and J-Y Ollitrault, Nucl. Phys. A 458, 745 (1986).
17. R. Landshoff, "A Numerical Method for Treating Fluid Flow in the Presence of Shocks", Los Alamos Report LA-1930 (1955).
18. M.L. Wilkins, J. Comp. Phys. 36, 281 (1980).
19. R. Christensen, in preparation.
20. W.A. Hiscock and L. Lindblom, Ann. Phys. 151, 466 (1983); W. Israel, Ann. Phys. 100, 310 (1976).
21. C. Eckart, Phys. Rev. 58, 919 (1940); L.D. Landau and E.M. Lifshitz, "Fluid Mechanics", Chapter 15 (1982, Pergamon Press).
22. L.P. Csernai and B. Lukacs, Acta Phys. Pol. B15, 149 (1984), M-C Chu, "One Dimensional Hydrodynamics of Ultra-Relativistic Heavy-Ion Collisions", in press, Phys. Rev. D.
23. N.L. Balàzs, B. Schürmann, K. Dietrich, and L.P. Csernai, Nucl. Phys. A424, 605 (1984).
24. Y. Kitazoe, M. Sano, Y. Yamamura, H. Furutani and K. Yamamoto, Phys. Rev C29, 828 (1984).
25. Kevin W. Thompson, J. Fluid Mech. 171, 365 (1986).
26. W.F. Noh, "Artificial Viscosity (Q) and Artificial Heat Flux (H) Errors for Spherically Divergent Shocks", UCRL-89623 (1983).
27. J. Centrella and J.R. Wilson, Ap. J. Suppl. 54, 229 (1984).
28. J.A. Zingman, T.L. McAbee, J.R. Wilson, C.T. Alonso, "Relativistic 3-D Nuclear Hydrodynamics with Monte Carlo Pions", UCRL-97153.
29. J.J. Molitoris, D. Hahn, C. Alonso, I. Collazo, P. D'Alessandris, T. McAbee, J. Wilson, J. Zingman, "Relativistic Nuclear Fluid Dynamics and VUU Kinetic Theory", UCRL-97270.
30. P. Danielewicz and G. Odyniec, Phys. Lett. 157B, 146(1985).